Weak Probabilistic Anonymity

Yuxin Deng, Catuscia Palamidessi, Jun Pang

Weak Probabilistic Anonymity*

Yuxin Deng[1]**, Catuscia Palamidessi[2], and Jun Pang[2]

[1] INRIA Sophia-Antipolis and Université Paris 7
[2] INRIA Futurs and LIX, École Polytechnique

Abstract. Anonymity means that the identity of the user performing a certain action is maintained secret. The protocols for ensuring anonymity often use random mechanisms which can be described probabilistically. In this paper we propose a notion of *weak* probabilistic anonymity, where *weak* refers to the fact that some amount of probabilistic information may be revealed by the protocol. This information can be used by an observer to infer the likeliness that the action has been performed by a certain user. The aim of this work is to study the degree of anonymity that the protocol can still ensure, despite the leakage of information.
We illustrate our ideas by using the example of the dining cryptographers with biased coins. We consider both the cases of nondeterministic and probabilistic users. Correspondingly, we propose two notions of weak anonymity and we investigate their respective dependencies on the biased factor of the coins.

1 Introduction

Anonymity is the property of keeping secret the identity of the user performing a certain action. The need for anonymity may raise in a wide range of situations, like postings on electronic forums, voting, delation, donations, and many others.

The protocols for ensuring anonymity often use random mechanisms. This is the case, for example, of the Dining Cryptographers [6], Crowds [10], and Onion Routing [14].

Various notions of probabilistic anonymity have been investigated in the literature [6, 10, 7, 4]. In this paper we propose a notion of *weak* probabilistic anonymity, where *weak* refers to the fact that some amount of probabilistic information may be revealed by the protocol. Typical causes may be either the presence of attackers which interfere with the normal execution the protocol, or some unavoidable imperfection of the internal mechanisms, or may even be inherent to the way the protocol is designed. In any case, the information leaked by the system can be used by an observer to infer the likeliness that the action has been performed by a certain user. The aim of this work is to study the degree of anonymity that the protocol can still ensure, despite the leakage of information.

We illustrate our ideas by using the example of the Dining Cryptographers Problem (DCP). In this protocol, a number of users (cryptographers) cooperate to ensure that the

* This work has been partially supported by the Project Rossignol of the ACI Sécurité Informatique (Ministère de la recherche et nouvelles technologies).
** Supported by the EU project PROFUNDIS.

occurrence of a certain action is made visible, while the cryptographer who has performed it remains anonymous. They achieve this goal by executing a certain algorithm which involves coin tossing. In the original formulation of [6] the coins are perfectly fair and no one (except the authorized cryptographers) gets any information about the results of the coins. As a consequence of these assumptions, the protocol ensures strong anonymity in the sense that, from the point of view of an observer, there is no way to infer that a cryptographers is more likely than another to have performed the action.

We consider a more realistic scenario in which some probabilistic information may be leaked by the system. In particular, we consider the case in which this happens due to imperfections in its internal mechanisms. In the case of the DCP, this means to relax the hypothesis of perfect fairness of the coins. It is worth noting that even if an observer does not know a priori whether and how much the coins in the DCP are biased, he may be able to infer it statistically by running the protocol several times [4]. One of the main purposes of this work is to investigate how the biased factor of the coins influences the level of anonymity that the system can still achieve.

An issue to consider when we deal with a probabilistic system is whether or not there is also some nondeterministic choice involved. Nondeterministic means that the choice is completely unpredictable. In anonymity protocols, the user which perform the action may be selected either nondeterministically or probabilistically. In the nondeterministic case, the probabilistic aspect of anonymity can only be relative to the probability of the observables, which derives solely from the randomness of the internal mechanisms of the protocol. The natural notion of anonymity is then that the probability of the observables does not give information about the user.

In the case of probabilistic users, there are two possible points of view under which one can define the notion of anonymity. Namely, we can focus on the probability of the observables, and require that they do not allow to infer information about the probability of the users (similarly to the nondeterministic case), or we can focus on the probability of the users, and require that the system does not allow to infer extra information about it through the observables. Interestingly, in the case of strong anonymity these two notions have been proved equivalent [4].

In this paper we consider both the cases of nondeterministic and probabilistic users, and we propose two notions of weak anonymity corresponding to the two points of view illustrated above. Although, as just said, in the limit case of strong anonymity these two notions are equivalent, their functional dependency on the biased factor of the coins turns out to be totally different.

1.1 Contributions

The main contributions of this work are:

- We propose two notions of weak probabilistic anonymity, for the cases of nondeterministic and probabilistic users, respectively.
- We consider the Dining Cryptographers with biased coins, and we study how the two notions of weak anonymity depend on the biased factor of the coins.
- We show how to code the formulas that expresses weak anonymity in PRISM, so that their validity can be checked automatically on a generic protocol.

1.2 Plan of the paper

In next section we recall some notions which are used in the rest of the paper: the Probabilistic Automata, the Dining Cryptographers Problem, and the framework for anonymity developed in [4]. In Section 3 we propose a notion of weak anonymity for nondeterministic users, and we study the dependency on the biased factor of the coins for the DCP. In Section 4 we do the same for the case of probabilistic users. In Section 5 we code in PRISM the DCP and the notions of anonymity. Finally, in Section 6 we conclude and discuss some related work.

2 Preliminaries

2.1 Nondeterminism and probability

In this paper we consider systems that can perform both probabilistic and nondeterministic choice. Intuitively, a probabilistic choice represents a set of alternative transitions, each of them associated to a certain probability of being selected. The sum of all probabilities on the alternatives of the choice must be 1, i.e. they form a *probability distribution*. Nondeterministic choice is also a set of alternatives, but we have no information on how likely one alternative is selected.

We take the point of view that a nondeterministic choice *is not a probabilistic choice with unknown probabilities*: in the latter, if we repeatedly run the program, we can infer the probability. For instance, if we have a choice between two transitions and we observe that they are selected with the same frequency, we can infer that the probability is close to $1/2$. In the nondeterministic case, this inference would be invalid. Nondeterministic means that the choice is totally unpredictable and that there is no assumption of regularity through time on the mechanisms that determine the selection.

There have been many models proposed in literature that combine both nondeterministic and probabilistic choice. One of the most general is the formalism of *probabilistic automata* proposed in [13]. We give here a brief and informal description of it.

A probabilistic automaton consists in a set of states, and labeled transitions between them. For each node, the outgoing transitions are partitioned in groups called *steps*. Each step represents a probabilistic choice, while the choice between the steps is nondeterministic.

Figure 1 illustrates some examples of probabilistic automata. We represent a step by putting an arc across the member transitions. For instance, in (a), state s_1 has two steps, the first is a probabilistic choice between two transitions with labels a and b, each with probability $1/2$. When there is only a transition in a step, like the one from state s_3 to state s_6, the probability is of course 1 and we omit it.

In this paper, we use only a simplified kind of automaton, in which from each node we have either a probabilistic choice or a nondeterministic choice (more precisely, either one step or a set of singleton steps), like in (b). In the particular case that the choices are all probabilistic, like in (c), the automaton is called *fully probabilistic*.

Given an automaton M, we denote by $etree(M)$ its unfolding, i.e. the tree of all possible executions of M (in Figure 1 the automata coincide with their unfolding because

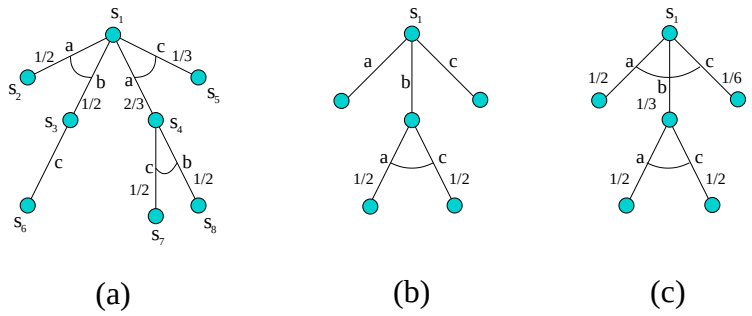

Fig. 1. Examples of probabilistic automata

there is no loop). If M is fully probabilistic, then each execution (maximal branch) of $etree(M)$ has a probability obtained as the product of the probability of the edges along the branch. In the finite case, we can define a probability measure for each set of executions, called *event*, by summing up the probabilities of the elements[3]. Given an event x, we will denote by $p(x)$ the probability of x. For instance, let the event c be the set of all computations in which c occurs. In (c) its probability is $p(c) = 1/3 \times 1/2 + 1/6 = 1/3$.

When nondeterminism is present, the probability can vary, depending on how we *resolve* the nondeterminism. In other words we need to consider a function ς that, each time there is a choice between different steps, selects one of them. By pruning the non-selected steps, we obtain a fully probabilistic execution tree $etree(M, \varsigma)$ on which we can define the probability as before. For historical reasons (i.e. since nondeterminism typically arises from the parallel operator), the function ς is called *scheduler*.

It should then be clear that the probability of an event is relative to the particular scheduler. We will denote by $p_\varsigma(x)$ the probability of the event x under the scheduler ς. For example, consider (a). We have two possible schedulers determined by the choice of the step in s_1. Under one scheduler, the probability of c is $1/2$. Under the other, it is $2/3 \times 1/2 + 1/3 = 2/3$. In (b) we have three possible schedulers under which the probability of c is 0, $1/2$ and 1, respectively.

2.2 The Dining Cryptographers

The general Dining Cryptographers Problem [6] is described as follows: A number of cryptographers, situated in the nodes of a given connected graph, are having a dinner. The representative of their organization (master) may or may not pay the bill of the dinner. If he does not, then he will select exactly one cryptographer and order him to pay the bill. The master will tell secretly each cryptographer whether he has to pay or not. The cryptographers would like to reveal whether the bill is paid by the master or

[3] In the infinite case things are more complicated: we cannot define a probability measure for all sets of execution, and we need to consider as event space the σ-field generated by the *cones* of $etree(M)$. However, in this paper, we consider only the finite case.

by one of them, but, in the latter case, they wish to keep anonymous the identity of the payer.

A possible solution to this problem, described in [6], is to associate a coin to each edge of the graph, visible only to the adjacent cryptographers. The coins are then tossed, and each cryptographer computes the binary sum of the adjacent coins (counting 0, say, for head and 1 for tail), adds 1 if he is the payer, and outputs the result.

In [6] it is proved that the payer is one of the cryptographers if and only if the binary sum of all the outputs is 1. Furthermore, if the coins are fair, then an external observer cannot identify the payer when it is one of the cryptographers.

The DCP will be a running example through the paper.

2.3 Anonymity systems

In this section we recall our approach to anonymity, as developed in [4].

We model the anonymity protocol as a probabilistic automaton M. The concept of anonymity is relative to the set of anonymous users and to what is visible to the observer. Hence, following [12, 11] we classify the actions of M into the three sets A, B and C as follows:

- A is the set of the anonymous actions $A = \{a(i) \,|\, i \in I\}$ where I is the set of the identities of the anonymous users and a is an injective functions from I to the set of actions, which we call *abstract action*. We also call the pair (I, a) *anonymous action generator*.
- B is the set of the observable actions. We will use b, b', ... to denote the elements of this set.
- C is the set of the remaining actions (which are unobservable).

Note that the actions in A normally are not visible to the observer, or at least, not for the part that depends on the identity i. However, for the purpose of defining and verifying anonymity we model the elements of A as visible outcomes of the system.

Definition 1. *An anonymity system is a tuple (M, I, a, B, Z, p), where M is a probabilistic automaton, (I, a) is an anonymous action generator, B is a set of observable actions, Z is the set of all possible schedulers for M, and for every $\varsigma \in Z$, p_ς is the probability measure on the event space generated by $etree(M, \varsigma)$.*

If the system is fully probabilistic, then Z is a singleton and we omit it.

We introduce the following notation to represent the events of interest:

- $a(i)$: all the executions in $etree(M, \varsigma)$ containing the action $a(i)$;
- a : all the executions in $etree(M, \varsigma)$ containing an action $a(i)$ for an arbitrary i;
- o : all the executions in $etree(M, \varsigma)$ containing as their maximal sequence of observable actions the sequence o (where o is of the form $b_1 b_2 \ldots b_n$ for some $b_1, b_2, \ldots, b_n \in B$). We denote by O (*observables*) the set of all such o's.

We use the symbols $\cup$, $\cap$ and $\neg$ to represent the union, the intersection, and the complement of events, respectively.

We wish to keep the notion of observables as general as possible, but we still need to make some assumptions on them. First, we want the observables to be disjoint events. Second, they must cover all possible outcomes. Third, an observable o must indicate unambiguously whether a has taken place or not, i.e. it either implies a, or it implies $\neg a$. In set-theoretic terms it means that either o is a subset of a or of the complement of a. Formally:

Assumption 1 (on the observables)

1. $\forall \varsigma \in Z. \forall o_1, o_2 \in O.\ o_1 \neq o_2 \ \Rightarrow\ p_\varsigma(o_1 \cup o_2) = p_\varsigma(o_1) + p_\varsigma(o_2)$
2. $\forall \varsigma \in Z.\ p_\varsigma(O) = 1$
3. $\forall \varsigma \in Z. \forall o \in O.\ (p_\varsigma(o \cap a) = p_\varsigma(o)) \ \vee \ p_\varsigma(o \cap \neg a) = p_\varsigma(o)$

Analogously, we need to make some assumption on the anonymous actions. We consider first the conditions tailored for the nondeterministic users: each scheduler determines completely whether an action of the form $a(i)$ takes place or not, and in the positive case, there is only one such i. Formally:

Assumption 2 (on the anonymous actions, for nondeterministic users)

$$\forall \varsigma \in Z.\ p_\varsigma(a) = 0 \vee (\exists i \in I.\ (p_\varsigma(a(i)) = 1\ \wedge\ \forall j \in I.\ j \neq i \Rightarrow p_\varsigma(a(j)) = 0))$$

In [4] the following strong notion of anonymity was proposed. Intuitively, given two schedulers ς and ϑ that both choose a (say $a(i)$ and $a(j)$, respectively), it should not be possible to detect from the probabilistic measure of the observables whether the scheduler was ς or ϑ (i.e. whether the selected user was i or j).

Definition 2 ((Strong) anonymity for nondeterministic users). *A system (M, I, a, B, Z, p) is anonymous if*

$$\forall \varsigma, \vartheta \in Z. \forall o \in O.\ p_\varsigma(a) = p_\vartheta(a) = 1\ \Rightarrow\ p_\varsigma(o) = p_\vartheta(o)$$

We now consider the case in which the users are fully probabilistic. The assumption on the anonymous actions in this case is much weaker: we only require that there be at most one user that performs a, i.e. $a(i)$ and $a(j)$ must be disjoint for $i \neq j$. Formally:

Assumption 3 (on the anonymous actions, for probabilistic users)

$$\forall i, j \in I.\ i \neq j\ \Rightarrow\ p(a(i) \cup a(j)) = p(a(i)) + p(a(j))$$

The probabilistic counterpart of Definition 2 can be formalized using the concept of *conditional probability*. Recall that, given two events x and y with $p(y) > 0$, the conditional probability of x given y, denoted by $p(x \,|\, y)$, is equal to $p(x \cap y)/p(y)$.

Definition 3. *A fully probabilistic system (M, I, a, B, p) is anonymous if*

$$\forall i, j \in I. \forall o \in O.\ (p(a(i)) > 0 \wedge p(a(j)) > 0) \Rightarrow p(o \,|\, a(i)) = p(o \,|\, a(j))$$

The notions of anonymity illustrated so far focus on the probability of the observables. In the case of probabilistic users, however, one can also approach the concept of anonymity from the point of view of the probabilistic information associated to the users. This is the perspective adopted in [7] to define what they call *conditional anonymity*. The idea is that a system is anonymous if the observations do not change the probability of the $a(i)$'s. In other words, we may know the probability of $a(i)$ by some means external to the system, but the system should not increase our knowledge about it. The same notion was proposed, implicitly, in [6]. This concept can be formulated in our framework as follows:

Definition 4 ((Strong) anonymity for probabilistic users). *A fully probabilistic system (M, I, a, B, p) is anonymous if*

$$\forall i \in I.\, \forall o \in O.\ \ p(o \cap a) > 0 \ \Rightarrow\ p(a(i) \,|\, o) = p(a(i) \,|\, a)$$

Despite Definitions 3 and 4 are based on conceptually different interpretations of anonymity, it was shown in [4] that they are equivalent.

The definitions of anonymity illustrated in this section are satisfied by the DCP only if the coins are fair. In next sections we propose weak versions of these definitions, which may be satisfied also when the coins are biased, depending on the biased factor.

3 Weak anonymity for nondeterministic users

In this section we propose a weak variant of Definition 2 and we study, in the particular case of the DCP, how this property depends on the biased factor of the coins.

Intuitively, the weakening consists in relaxing the constraint that the probability of an observer implying a is the same under every scheduler. Instead, we require that the difference between any two such probabilities does not exceed a certain parameter α. Formally:

Definition 5 (α-anonymity for nondeterministic users). *Given $\alpha \in [0, 1]$, a system (M, I, a, B, Z, p) is α-anonymous if*

$$max\{\ p_\varsigma(o) - p_\vartheta(o)\ \mid \varsigma, \vartheta \in Z,\ o \in O,\ p_\varsigma(o \cap a) = p_\varsigma(o),\ p_\vartheta(o \cap a) = p_\vartheta(o)\} = \alpha$$

Intuitively, $p_\varsigma(o) - p_\vartheta(o) \ = \ \alpha$ means that, whenever we observe o, we suspect that user i is more likely than user j to have performed the action by an additive factor α (where i and j represent the users selected by ς and ϑ, respectively).

Let us consider the DCP on a linear graph consisting of three nodes, i.e. three cryptographers $Crypt_0$, $Crypt_1$, and $Crypt_2$, and two edges, $Coin_0$ between $Crypt_0$ and $Crypt_1$, and $Coin_1$, between $Crypt_1$ and $Crypt_2$.

In case one of the cryptographers pays (event a), the possible observables are

$$o_1 = 111 \qquad o_2 = 100 \qquad o_3 = 010 \qquad o_4 = 001$$

where $b_0 b_1 b_2$ refers to the outputs of $Crypt_0$, $Crypt_1$ and $Crypt_2$, respectively. For instance, if $Crypt_1$ is the payer, then o_1 is obtained when both the two coins give 1, o_2

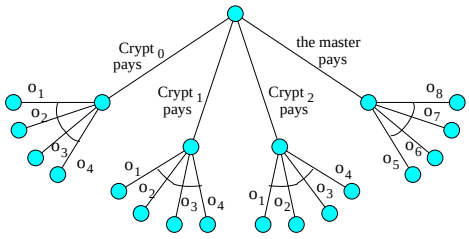

Fig. 2. The DCP with three cryptographers and nondeterministic master.

is obtained when $Coin_0$ gives 1 and $Coin_1$ gives 0, etc. In case the master pays, then the possible observables are $o_5 = 110, o_6 = 101, o_7 = 011, o_8 = 000$. For instance, $o_5 = 110$ is obtained when $Coin_0$ gives 1 and $Coin_1$ gives 0.

The probabilistic automaton corresponding to this situation is illustrated in Figure 2. For simplicity, we have drawn only the "big-step-transitions" corresponding to the observables o_1, o_2 etc. They represents sequences of "small-step-transitions" where each coin is flipped, then each cryptographer in turn reads the coins, then it computes and output the results.

It is important to note that we will consider only one form of nondeterminism: that associated to the choice of the master (*nondeterministic master*, which in the DCP is synonymous of nondeterministic users). In general in a system there is also the nondeterminism caused by the different possible interleaving of the various components of the system, but here, for simplicity, we will assume that the order in which the various components of the system (master, cryptographers, coins) execute their operations is fixed. In any case, it can be shown that this latter form of nondeterminism would not affect the properties of the DCP with respect to anonymity.

Let us represent by β_i the "biased factor" of $Coin_i$, i.e. the probability that $Coin_i$ gives 0. We want to determine how the parameter α of anonimity (Definition 5) depends on β_0 and β_1.

Consider, for each scheduler that selects a payer among the cryptographers, the possible observables and their probability measure. A simple calculation gives the figures shown in Table 1. Then by case analysis, we obtain:

$$\alpha = \begin{cases} |1 - (\beta_0 + \beta_1)| & \text{if } (\beta_0, \beta_1 \leq 0.5) \text{ or } (\beta_0, \beta_1 \geq 0.5); \\ |\beta_0 - \beta_1| & \text{if } (\beta_0 > 0.5 \text{ and } \beta_1 < 0.5) \text{ or } (\beta_0 < 0.5 \text{ and } \beta_1 > 0.5); \end{cases}$$

Figure 3 shows the graph of α as a function of β_0 and β_1.

The above analysis can be extended to the general case of linear graphs with any number of nodes.

Theorem 1. *In the DCP on a linear graph with n nodes the α in Definition 5 depends on the β_i's as follows:*

$$\alpha = \prod_{\beta_i \geq 0.5} \beta_i \prod_{\beta_j < 0.5} (1 - \beta_j) \; - \; \prod_{\beta_i \geq 0.5} (1 - \beta_i) \prod_{\beta_j < 0.5} \beta_j$$

Observables: $o_1 = 111$, $o_2 = 100$, $o_3 = 010$, $o_4 = 001$			
	Crypt$_0$ pays	*Crypt*$_1$ pays	*Crypt*$_2$ pays
$p(o_1)$	$\beta_0(1-\beta_1)$	$(1-\beta_0)(1-\beta_1)$	$(1-\beta_0)\beta_1$
$p(o_2)$	$\beta_0\beta_1$	$(1-\beta_0)\beta_1$	$(1-\beta_0)(1-\beta_1)$
$p(o_3)$	$(1-\beta_0)\beta_1$	$\beta_0\beta_1$	$\beta_0(1-\beta_1)$
$p(o_4)$	$(1-\beta_0)(1-\beta_1)$	$\beta_0(1-\beta_1)$	$\beta_0\beta_1$

Table 1. Probabilities of the observables in the case of 3 cryptographers on a linear graph.

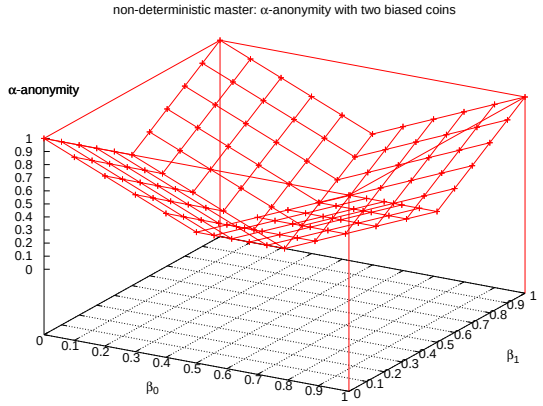

Fig. 3. The dependency of α-anonymity on β_0 and β_1 in the case of three cryptographers.

Proof The highest possible probability for an observable corresponds to the coin configuration

$$Coin_i = 0 \text{ for } \beta_i \geq 0.5 \quad \text{and} \quad Coin_j = 1 \text{ for } \beta_j < 0.5 \tag{1}$$

Conversely, the minimal probability corresponds to

$$Coin'_i = 1 \text{ for } \beta_i \geq 0.5 \quad \text{and} \quad Coin'_j = 0 \text{ for } \beta_j < 0.5 \tag{2}$$

Clearly these configurations are obtained, respectively, with probabilities

$$p_1 = \prod_{\beta_i \geq 0.5} \beta_i \prod_{\beta_j < 0.5} (1-\beta_j) \quad \text{and} \quad p_2 = \prod_{\beta_i \geq 0.5} (1-\beta_i) \prod_{\beta_j < 0.5} \beta_j$$

we only need to show, now, that both these probabilities can be obtained (under different schedulers) for the same observable.

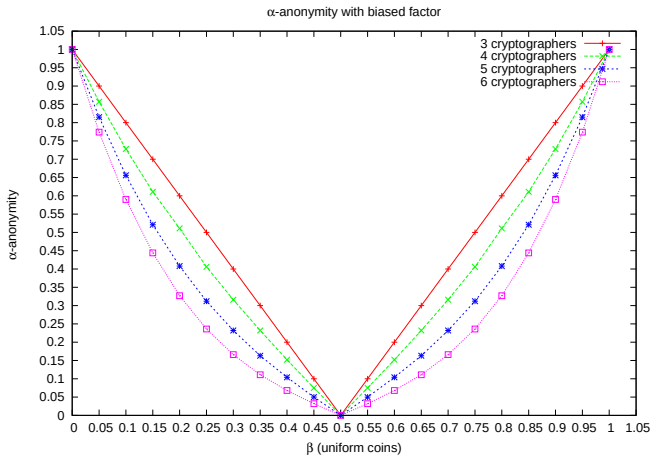

Fig. 4. The dependency of α-anonymity on β's in the case of 3-6 cryptographers.

Consider the coin configuration in (1). Let ς be the scheduler that selects $Crypt_0$ as the payer. Then the system will output the observable $o = b_0 b_1 \dots b_{n-1}$ where (using $\oplus$ to represent the binary sum)

$$\begin{aligned} b_0 &= Coin_0 \oplus 1 \\ b_i &= Coin_{i-1} \oplus Coin_i \qquad \text{for } 1 \leq i \leq n-2 \\ b_{n-1} &= Coin_{n-2} \end{aligned}$$

Clearly $p_\varsigma(o) = p_1$.

Consider now the coin configuration in (2). Let ϑ be the scheduler that selects $Crypt_{n-1}$ as the payer. It is easy to see that the output $o' = b'_0 b'_1 \dots b'_{n-1}$ of the system is the same as before, in fact for each i, $Coin'_i = Coin_i \oplus 1$. Hence we have:

$$\begin{aligned} b'_0 &= Coin'_0 = Coin_0 \oplus 1 = b_0 \\ b'_i &= Coin'_{i-1} \oplus Coin'_i = Coin_{i-1} \oplus 1 \oplus Coin_i \oplus 1 \\ &= Coin_{i-1} \oplus Coin_i = b_i \qquad \text{for } 1 \leq i \leq n-2 \\ b'_{n-1} &= Coin'_{n-2} \oplus 1 = Coin_{n-2} \oplus 1 \oplus 1 = Coin_{n-2} = b_{n-1} \end{aligned}$$

Hence we have $o' = o$ and $p_\vartheta(o) = p_2$. □

It is possible to show that the above theorem holds also when the topology is a ring. On the other hand, it does not hold for graphs which contain one or more nodes with an odd number of adjacent edges.

Figure 4 illustrates the dependency of α on β for three to six cryptographers, where for all i, $\beta_i = \beta$ (uniform coins). We note that the anonymity level increases (i.e. α

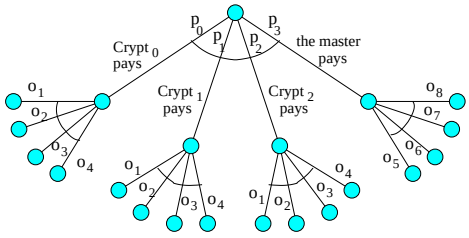

Fig. 5. The DCP with three cryptographers and probabilistic master.

decreases) as the number of cryptographers increases. If the coins are fair ($\beta = 0.5$), then we have strong anonymity, i.e. $\alpha = 0$. In the two extreme cases of $\beta = 0$ or $\beta = 1$, the α-anonymity is always 1, which is maximal. It is also possible to show that α is expressed by a polynomial on β whose degree is $n - 1$ if n is even, and $n - 2$ if n is odd.

4 Weak anonymity for probabilistic users

In this section we consider the case in which the user is selected according to a certain probability distribution. Since we assume that we have no other source of nondeterminism, the automata that we consider in this section are fully probabilistic. For example, in the case of the DCP with three cryptographers, we have the automaton represented in Figure 5.

4.1 Focusing on the probabilities of the observables

The fully probabilistic version of Definition 5, corresponding also to the weak version of Definition 3, is the following:

Definition 6 (α-anonymity for probabilisitic users). *Given $\alpha \in [0, 1]$, a fully probabilistic system (M, I, a, B, p) is α-anonymous if*

$$max\{\, p(o \mid a(i)) - p(o \mid a(j)) \;\mid i, j \in I,\; o \in O,\; p(a(i)) > 0,\; p(a(j)) > 0 \,\} = \alpha$$

Like Definition 5, this notion focuses on the probability of the observables.

It is possible to prove that for the DCP the α of the above definition depends on the β_i's exactly like the α for nondeterministic users (Definition 5). In fact consider, in the case of nondeterministic users, a scheduler ς that selects i, i.e. $p_\varsigma(a(i)) = 1$. Then assume, in the case of probabilistic users, that $p_\varsigma(a(i)) > 0$. It is easy to see that $p_\varsigma(o) = p(o \mid a(i))$.

So, in a sense, the notion of anonymity proposed in Definition 6 does not seem to introduce any new technical challenge with respect to the study done for the case of nondeterministic users.

In next section we investigate, instead, the weak version of the alternative notion of anonymity given in Definition 4.

4.2 Focusing on the probabilities of the users

We take here the point of view that anonymity means to preserve the probability of the users, like in [6] and [7].

Definition 7 (α-anonymity for probabilisitic users – alternative notion). *Given $\alpha \in [0,1]$, a fully probabilistic system (M, I, a, B, p) is α-anonymous if*

$$max\{\, p(a(i) \mid o) - p(a(i) \mid a) \;\mid i \in I,\; o \in O,\; p(o \cap a) > 0\} = \alpha$$

Intuitively, $p(a(i) \mid o) - p(a(i) \mid a) = \alpha$ means that, after observing o, the probability we attribute to i as the performer of the action, has increased by an additive factor α.

We study now the dependency of α on the β_i's in the case of the DCP with n cryptographers on a linear graph. We need to introduce some definitions: Let p_i be the probability that $Crypt_i$ is the payer. Of course, the probability that one of the cryptographers is the payer is then $\sum_{i=0}^{n-1} p_i$. Let k be the index of the cryptographer with the highest probability, i.e.

$$p_k = max\{p_i \mid i \in [0, n-1]\}$$

For $i \in [0, n-2]$, define

$$\gamma_i = \begin{cases} \beta_i & \text{if } \beta_i \geq 0.5 \\ 1 - \beta_i & \text{otherwise} \end{cases}$$

Finally, for an arbitrary $j \in [0, n-1]$, define

$$q_j = \begin{cases} \displaystyle\prod_{i=0}^{j-1} \gamma_i \prod_{i=j}^{k-1} (1-\gamma_i) \prod_{i=k}^{n-2} \gamma_i & \text{if } j \leq k \\ \displaystyle\prod_{i=0}^{k-1} \gamma_i \prod_{i=k}^{j-1} (1-\gamma_i) \prod_{i=j}^{n-2} \gamma_i & \text{otherwise} \end{cases}$$

We are now ready to show how α depends on the β_i's:

Theorem 2. *In the DCP on a linear graph with n nodes the α in Definition 7 depends on the β_i's (and on the p_i's) as follows:*

$$\alpha = \frac{q_k\, p_k}{\displaystyle\sum_{j=0}^{n-1} q_j\, p_j} \;-\; \frac{p_k}{\displaystyle\sum_{j=0}^{n-1} p_j} \tag{3}$$

Proof By definition, the configuration of the coins with the highest probability is the one in which $Coin_i = 0$ if $\beta_i \geq 0.5$ and $Coin_i = 1$ otherwise. The probability of this configuration is

$$\prod_{\beta_i \geq 0.5} \beta_i \prod_{\beta_\ell < 0.5} (1 - \beta_\ell) \;=\; \prod_{i=0}^{n-2} \gamma_i = q_k$$

Consider now the event $a(k)$ expressing that $Crypt_k$ is the payer, and let $o = b_0 b_1 \ldots b_{n-1}$ be the observable which corresponds to the above coin configuration in combination with the event $a(k)$. We will show that o and $a(k)$ maximize the expression $p(a(i) \mid o') - p(a(i) \mid a)$ and that it is equal to the Formula (3). First we need to compute the conditional probability $p(a(k) \mid o) = p(o \cap a(k))/p(o)$. By definition, $p(o \cap a(k)) = q_k\ p_k$. As for $p(o)$, observe that $p(o \cap a(j)) = q_j\ p_j$ for any $j \in [0, n-1]$, in fact to obtain the same o when $Crypt_j$ is the payer, it is sufficient to flip all the coins between j and k, which gives a coin configuration with probability q_j. Hence, we have

$$p(o) = \sum_{j=0}^{n-1} p(o \cap a(j)) = \sum_{j=0}^{n-1} q_j\ p_j$$

Finally it is easy to see that $p(a(k) \mid o)$ maximizes $p(a(i) \mid o')$, that it is linear on $p(a(k)$, and that $p(a(j) \mid a) = p(a(j))/p(a)$. Hence, $p(a(k) \mid o) - p(a(k) \mid a)$ maximizes $p(a(i) \mid o') - p(a(i) \mid a)$ and coincides with the Formula (3). □

Figure 6 shows the dependency of α on the β_i's in the case of three cryptographers. The various graphs refer to different probability distributions for the payer. It is worth noting that, in contrast to the notion of α-anonymity given in Definition 6, the version presented in this section depends not only the β_i's, but also on the $p(a(i))$'s. On the other hand, in the limit case of strong anonymity, the two notions are equivalent, as explained in Section 2.3.

5 Automatic Analysis

In the case of a very simple topology (linear graphs) we have been able to express the dependency of α on the β_i's with a mathematical formula. In this way, if we have a system whose internal bias are known, it is immediate to check whether it satisfies weak anonymity (for a given α) or not. It is possible to extend the method also to rings, but as the graphs get more complicated, it is not clear how to proceed to find the formula that express the dependency. This is a typical situation for most real-life systems: the symbolic analysis is often unfeasible, and we have to resort to automatic tools supported by computers.

In this section, we describe how to use the probabilistic model checker PRISM to check the property of the α-anonymity for the DCP. We consider both nondeterministic and probabilistic masters (recall that in the DCP nondeterministic/probabilistic master is synonymous of nondeterministic/probabilistic users). We model the DCP as a discrete-time Markov chain (DTMC) in the case of a probabilistic master, and as Markov decision process (MDP) in the case of a nondeterministic master[4]. The PRISM input language is a simple, state-based language, based on the Reactive Modules formalism of Alur and Henzinger [1]. The events are formalized using the temporal probabilistic logic PCTL [8]. Once this translation is done, we can use PRISM to compute

[4] DTMC and MDP, which are the formats accepted by PRISM, can be seen as special cases of probabilistic automata.

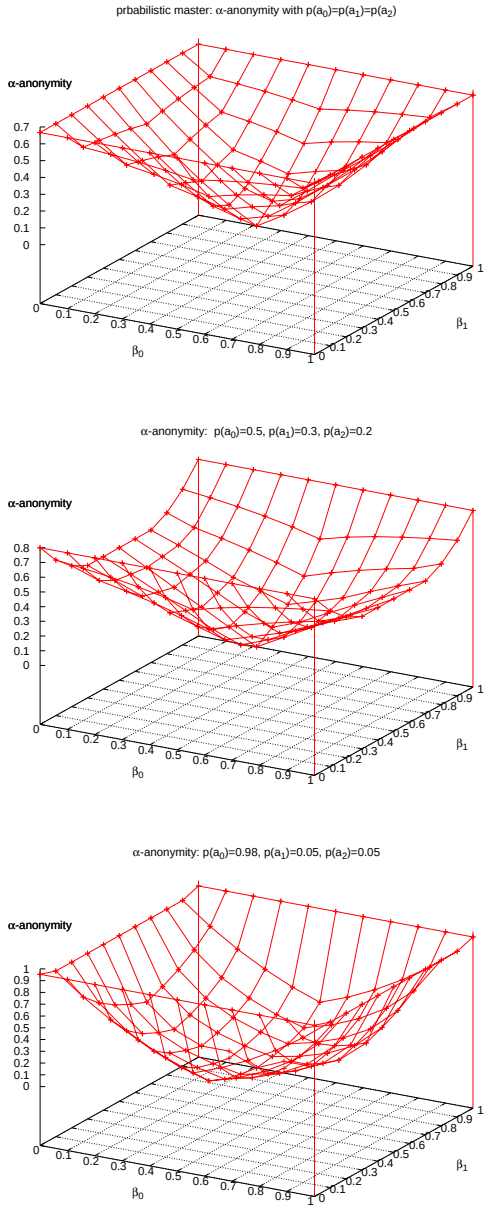

Fig. 6. The dependency of α-anonymity on the β_i's in the case of three cryptographers.

the probabilities of the relevant events so to check α-anonymity. A brief overview of PRISM and PCTL is given in the appendix.

The following code is for three cryptographers and two coins arranged in a line. It can be easily generalized to more cryptographers and a different graph structure. First we describe the variables we use in the model. N is the number of cryptographers, and, since the topology is a line, there are N-1 coins in the model. The probabilities of each coin of showing head are defined as beta0, beta1, etc. We define three more state variables: s_master: [0..2] for the master, s_coin: [0..N-1] to indicate how many coins have been flipped, and s_crypt: [0..N] to indicates how many cryptographers have decided their outputs. payerid=N indicates that either no cryptographer will pay or the master hasn't decided yet.. Initially, they are all 0. Once the execution terminates, we will have s_master=2, s_coin=N-1, and s_crypt=N. The variable payerid: [0..N] init N is used to record who is the payer. The variable toss: bool init false is used to let the coins to be flipped after the master has made his decision.

In the following, we consider the case $N = 3$. We use the variables crypt0, crypt1, crypt2 to record the values computed by each cryptographer, that are either 0 or 1 and depend on whether the cryptographer is paying and on the sides of the coins the cryptographer can see. Initially, their values are 0. In the model, there are two coins coin0 and coin1. The first is shared shared by Cryptographers 0 and 1, the second is shared by Cryptographers 1 and 2. We use 0 for head, and 1 for tail.

Next, we describe the behavior of the master, the coins and the cryptographers. If the master is nondeterministic, he will decide nondeterministically the payer: one of the cryptographers (payerid = 0, 1, or 2) or himself (payerid = 3). Once he has made the decision, the value of toss is set to true, in order to let the coins to be flipped.

```
[] (s_master=0) → (s_master'=1) & (payerid'=0);
[] (s_master=0) → (s_master'=1) & (payerid'=1);
[] (s_master=0) → (s_master'=1) & (payerid'=2);
[] (s_master=0) → (s_master'=1) & (payerid'=3);
[] (s_master=1) & (!toss) → (s_master'=2) & (toss'=true);
```

If the master is probabilistic, then the choice of the payer is based on a probability distribution. For instance:

```
[] (s_master=0) →
                0.5: (s_master'=1) & (payerid'=0) +
                0.3: (s_master'=1) & (payerid'=1) +
                0.1: (s_master'=1) & (payerid'=2) +
                0.1: (s_master'=1) & (payerid'=3);
[] (s_master=1) & (!toss) → (s_master'=2) & (toss'=true);
```

Once toss becomes true, the coins start to flip. With probabilities beta0 and beta1, the side of the coins will be head. With probabilities 1-beta0 and 1-beta1, the side of the coins will be tail. Each time when a coin is flipped, the value of s_coin is increased by one.

```
[] (s_coin=0) & (toss) →
                beta0: (coin0'=0) & (s_coin'=s_coin+1) +
```

```
                    (1-beta0): (coin0'=1) & (s_coin'=s_coin+1);
[] (s_coin=1) & (toss) →
                    beta1: (coin1'=0) & (s_coin'=s_coin+1) +
                    (1-beta1): (coin1'=1) & (s_coin'=s_coin+1);
```

After all the coins have been flipped (s_coin=N-1), the cryptographers calculate the value of their variable crypt0, crypt1 and crypt2. Once a cryptographer has terminated this calculation, the value of s_crypt is increased by 1. Since the Cryptographers 0 and 2 sit at the two ends of the line, they can only observe one coin: Cryptographer 0 sees Coin 0, and Cryptographer 2 sees Coin 1. If Cryptographer 0 is the payer, he will set the variable crypt0 to 1 if he sees the head of Coin 0, and to 0 otherwise. If Cryptographer 0 is not the payer, he will set crypt0 to 0 if he sees the head of Coin 0, and to 1 otherwise. The code for Cryptographer 2 is similar: just rename crypt0 into crypt2, coin0 into coin1, and s_crypt=0 into s_crypt=2.

```
[] (s_crypt=0) & (s_coin=N-1) & (payerid=0) & (coin0=0) s→
                    (crypt0'=1) & (s_crypt'=s_crypt+1);
[] (s_crypt=0) & (s_coin=N-1) & !(payerid=0) & (coin0=0) →
                    (crypt0'=0) & (s_crypt'=s_crypt+1);
[] (s_crypt=0) & (s_coin=N-1) & (payerid=0) & (coin0=1) →
                    (crypt0'=0) & (s_crypt'=s_crypt+1);
[] (s_crypt=0) & (s_coin=N-1) & !(payerid=0) & (coin0=1) →
                    (crypt0'=1) & (s_crypt'=s_crypt+1);
```

The behavior of Cryptographer 1 is slightly different, since he can observe two coins. If he is the payer, he will set the variable crypt1 to 1 if the two coins have the same side, and to 0 otherwise. If he is not the payer, he will set crypt1 to 0 if the two coins have the same side, and to 1 otherwise.

```
[] (s_crypt=1) & (s_coin=N-1) & (payerid=1) & (coin1=coin0) →
                    (crypt1'=1) & (s_crypt'=s_crypt+1);
[] (s_crypt=1) & (s_coin=N-1) & !(payerid=1) & (coin1=coin0) →
                    (crypt1'=0) & (s_crypt'=s_crypt+1);
[] (s_crypt=1) & (s_coin=N-1) & (payerid=1) & !(coin1=coin0) →
                    (crypt1'=0) & (s_crypt'=s_crypt+1);
[] (s_crypt=1) & (s_coin=N-1) & !(payerid=1) & !(coin1=coin0) →
                    (crypt1'=1) & (s_crypt'=s_crypt+1);
```

A self-loop is added in the end of the specification to avoid deadlock states[5].

```
[] (s_coin=N-1) & (s_master=2) & (s_crypt=N) →
                    (s_coin'=N-1) & (s_master'=2) & (s_crypt'=N);
```

In the DCP, an external observer can see the values of the variables crypt0, crypt2 and crypt2. Furthermore, the values of the variables in the PRISM model define the states of the system. For example, the following predicate represents the final states in which all cryptographers output 1. We denote it by o_1.

[5] This is required by the design of PRISM.

(crypt0=1) & (crypt1=1) & (crypt2=1) &
(s_crypt=3) & (s_coin=2) & (s_master=2)

For each type of master (nondeterministic or probabilistic) we can describe observables as a PCTL formula by using the $\mathcal{P}$ operator (see Appendix A.2). Then, we can use PRISM to compute the probability of each observable for the analysis of α-anonymity.

Nondeterministic master: If the master is nondeterministic, we can compute the maximum and the minimum probability of each observable, under any possible scheduler that selects one of the cryptographers to pay. Below, we specify the PCTL formulas to compute the probabilities of observable o_1.

$$\mathcal{P}_{max=?}[true\,\mathcal{U}\ o_1] \qquad \text{and} \qquad \mathcal{P}_{min=?}[true\,\mathcal{U}\ o_1]$$

Thus, it is sufficient to use the formulation of α-anonymity given by the following proposition, whose proof is immediate:

Proposition 1. *A system (M, I, a, B, Z, p) is α-anonymous (with respect to nondeterministic users) if*

$$\begin{array}{l} max\{\ max\{\ p_\varsigma(o)\ \mid \varsigma \in Z,\ p_\varsigma(o \cap a) = p_\varsigma(o)\ \} \\ \qquad - \\ \qquad min\{\ p_\vartheta(o)\ \mid \vartheta \in Z,\ p_\vartheta(o \cap a) = p_\vartheta(o)\ \}\ \mid\ o \in O\ \} \ = \ \alpha \end{array}$$

The results in Figure 4 have been checked using PRISM.

Probabilistic master: When the master is nondeterministic, α-anonymity is defined as

$$max\{\ p(a(i) \mid o) - p(a(i) \mid a)\ | i \in I,\ o \in O,\ p(o \cap a) > 0\ \} = \alpha.$$

Since PRISM does not support the calculation of conditional probability as a primitive, we have to compute each $p(a(i) \mid o)$ using the equivalent expression $p(a(i) \cap o)/p(o)$. As for $p(a(i) \mid a)$, this is the same as $p(a(i))/p(a)$. For example, in case of three cryptographers, $p(a(0) \cap o_1)$ can be computed by using the PCTL formula

$$\mathcal{P}_{=?}[true\,\mathcal{U}\ o_1 \wedge (payerid = 0)]$$

The results presented in Figure 6 have been checked using PRISM.

6 Conclusion

We propose two notions of weak probabilistic anonymity, for the cases of nondeterministic and probabilistic users, respectively. We have applied these two notions to the DCP with biased coins, and we have described the functional dependency of the weakness level on the biased factor of the coins. Furthermore we have coded in PRISM the DCP and the formulas that express weak anonymity.

This paper builds on the framework of probabilistic anonymity proposed in [4] that we have summarized in Section 2.3. The notions that we investigate here represent a generalization of the strong probabilistic anonymity proposed in [4].

To our knowledge, the first notion of probabilistic anonymity was proposed (although not with an explicit definition) in [6]. That notion corresponds to one of the notions of strong anonymity for probabilistic users investigated in [4], and more precisely, to the one recalled in Definition 4. This is the notion for which we have given the weak version in Definition 7.

Reiter and Robin have proposed in [10] an hierarchy of notions of probabilistic anonymity, at different levels of strength:

Beyond suspicion The actual user (i.e. the user that performed the action) is not more likely (to have performed the action) than every other user.

Probable innocence The actual user has probability less than $1/2$.

Possible innocence There is a non trivial probability that another used could have performed the action.

These notions were only given informally in [10]. If one has to interpret them literally, they do not have much in common with the notions investigated here (and in [4]). However, we suspect that the authors intended to express notions similar to ours. In fact, the property of anonymity they prove for the system Crowds (probable innocence) does not mention at all the probability of the user, but only the probability of the observables. A sort of probable innocence in that sense could be expressed formally by our Definition 5 by taking $\alpha = 1/2$.

Halpern and O'Neill have proposed in [7] various notions of probabilistic anonymity, focusing on the probability of the users. Their principal notion is based on epistemic logic and is formulated as a requirement on the knowledge of the observer about the probability of the user. They have given both strong and weak version of this notion, capturing formally the three levels of the hierarchy proposed by [10] (see above). Again, these notions do not seem directly related to the ones we investigate in this paper. On the other hand, Halpern and O'Neill have proposed also another notion, called *conditional anonymity* (cfr. Definition 4.4 in [7]), which corresponds to the strong probabilistic anonymity recalled in Definition 4.

References

1. R. Alur and T.A. Henzinger. Reactive modules. *Formal Methods in System Design*, 15(1):7–48, 1999.
2. C. Baier, B. Haverkort, H. Hermanns, and J.-P. Katoen. Model checking continuous-time markov chains by transient analysis. In *Proceedings of the 12th Conference on Computer Aided Verifi cation*, volume 1855 of *Lecture Notes in Computer Science*, pages 358–372. Springer-Verlag, 2000.
3. C. Baier and M. Kwiatkowska. Model checking for a probabilistic branching time logic with fairness. *Distributed Computing*, 11(3):125–155, 1998.
4. Mohit Bhargava and Catuscia Palamidessi. Probabilistic anonymity. Technical report, INRIA Futurs and LIX, 2005. Submitted for publication. `http://www.lix.polytechnique.fr/~catuscia/papers/Anonymity/report.ps`.
5. A. Bianco and L. de Alfaro. Model checking of probabilistic and nondeterministic systems. In *Proceedings of the 15th Conference on Foundations of Software Technology and Theoretical Computer Science*, volume 1026 of *Lecture Notes in Computer Science*, pages 499–513. Springer-Verlag, 1995.

6. David Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1:65–75, 1988.
7. Joseph Y. Halpern and Kevin R. O'Neill. Anonymity and information hiding in multiagent systems. In *Proc. of the 16th IEEE Computer Security Foundations Workshop*, pages 75–88, 2003.
8. H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6(5):512–535, 1994.
9. M.Z. Kwiatkowska, G. Norman, , and D. Parker. PRISM: Probabilistic symbolic model checker. In *Proceedings of the 12th Conference on Computer Performance Evaluation, Modelling Techniques and Tools*, volume 2324 of *Lecture Notes in Computer Science*, pages 200–204. Springer-Verlag, 2002.
10. Michael K. Reiter and Aviel D. Rubin. Crowds: anonymity for Web transactions. *ACM Transactions on Information and System Security*, 1(1):66–92, 1998.
11. Peter Y. Ryan and Steve Schneider. *Modelling and Analysis of Security Protocols*. Addison-Wesley, 2001.
12. Steve Schneider and Abraham Sidiropoulos. CSP and anonymity. In *Proc. of the European Symposium on Research in Computer Security (ESORICS)*, volume 1146 of *Lecture Notes in Computer Science*, pages 198–218. Springer-Verlag, 1996.
13. Roberto Segala and Nancy Lynch. Probabilistic simulations for probabilistic processes. *Nordic Journal of Computing*, 2(2):250–273, 1995. An extended abstract appeared in *Proceedings of CONCUR '94*, LNCS 836: 481-496.
14. P.F. Syverson, D.M. Goldschlag, and M.G. Reed. Anonymous connections and onion routing. In *IEEE Symposium on Security and Privacy*, pages 44–54, Oakland, California, 1997.

A Appendix

A.1 A brief overview of PRISM

PRISM [9] is a probabilistic model checker. It allows one to model and analyze systems and algorithms containing probabilistic aspects. PRISM supports three kinds of probabilistic models: discrete-time Markov chains (DTMCs), Markov decision processes (MDPs) and continuous-time Markov chains (CTMCs). We present the first two models briefly. A more detailed description of each model and PRISM can be found at http://www.cs.bham.ac.uk/ dxp/prism/.

A DTMC can be defined as consisting of a finite set of states S, an initial state s_0, a transition probability matrix $P : S \times S \to [0, 1]$ such that $\forall s \in S, \sum_{s' \in S} P(s, s) = 1$, and a labeling function from states to a finite set of atomic predicates $L : S \to 2^{AP}$. MDPs extend DTMCs by allowing both probabilistic and nondeterministic behavior. An MDP is defined as consisting of a set of states S, an initial state s_0, a function $Steps$ which maps each state in S to a finite non-empty set of probability distributions over S, and a labeling function L. The transition from a state $s \in S$ is determined by selecting an element μ of ***Steps(s)*** nondeterministically and then choosing a state probabilistically, according to the distribution μ.

A system in PRISM is composed of a number of modules that contain local variables, and that can interact with each other. The behavior of a DTMC is described by a set of commands of the form:

$$[a]\ g \to \lambda_1 : u_1 + \ldots + \lambda_\ell : u_\ell;$$

a is an action label in the style of process algebras, which introduces synchronization into the model. It can only be performed simultaneously by all modules that have an occurrence of action label a in their specification. If a transition does not have to synchronize with other transitions, then no action label needs to be provided for this transition. The symbol g is a predicate over all the variables in the system. Each u_i describes a transition which the module can make if g is true. A transition updates the value of the variables by giving their new *primed* value with respect to their *unprimed* value. The λ_i are used to assign probabilistic information to the transition. It is required that $\lambda_1 + \cdots + \lambda_\ell = 1$. This probabilistic information can be omitted if $\ell = 1$ (and so $\lambda_1 = 1$). PRISM considers states without outgoing transitions as error states; terminating states can be modeled by adding a self-loop. PRISM models which are MDPs can also exhibit *local non-determinism*, which allows the modules to make nondeterministic choices themselves. For example, the probabilistic choice in the previous command can be made nondeterministic as follows:

$[a]\ g \rightarrow u_1;$
...
$[a]\ g \rightarrow u_\ell;$

A.2 A brief overview of PCTL

PRISM performs model checking against specifications written in the probabilistic temporal logic PCTL [8, 5, 3] if the model is a DTMC or an MDP, or CSL [2] in the case of a CTMC. PCTL can express properties of the form "under any scheduling of processes, the probability that event E occurs is at least p". The syntax of PCTL is given as follows:

$$\Phi ::= true \mid false \mid a \mid \Phi \wedge \Phi \mid \Phi \vee \Phi \mid \neg\Phi \mid \mathcal{P}_{\bowtie p}[\Psi]$$

$$\Psi ::= X\Phi \mid \Phi\, \mathcal{U}^{\leq k}\, \Phi \mid \Phi\, \mathcal{U}\, \Phi$$

where a are predicates over state variables, $\bowtie \in \{<, \leq, \geq, >\}$ is a Boolean operator, $p \in [0, 1]$ is a probability and k is an integer. Φ denotes a *state formula* and Ψ a *path formula*, these are evaluated over states and paths of a DTMC or MDP, respectively, where a path is a sequence of states connected by transitions. A state always satisfies $true$, it never satisfies $false$. The Boolean operators have the usual meanings. $X\Phi$ is true if and only if Φ is satisfied in the next state of the path. $\Phi_1\, \mathcal{U}^{\leq k}\, \Phi_2$ is true if and only if Φ_2 is satisfied in one of the first k states in the path and Φ_1 is satisfied in all preceding states. $\Phi_1\, \mathcal{U}\, \Phi_2$ is true if and only if $\Phi_1\, \mathcal{U}^{\leq k}\, \Phi_2$ for some $k \geq 0$. The formula $\mathcal{P}_{\bowtie}[\Psi]$ is true in a state s if the probability that a path starting in s satisfying the path formula Ψ meets the bound $\bowtie p$. The definition of a probability measure over paths of a DTMC is standard [8]. For MDPs, a probability measure can only be defined once the nondeterministic choices have been removed. Hence, a more accurate interpretation of the formula $\mathcal{P}_{\bowtie p}[\Psi]$ is that the probability of Ψ being satisfied meets the bound $\bowtie p$ for all resolutions of non-determinism [3].

For the purpose of the analysis of α-anonymity, we are interested in formulas of the form $\mathcal{P}_{\bowtie p}[true\ \mathcal{U}\ \Phi]$, evaluated in the initial state s_0. Here, Φ specified a system configuration of interest, typically representing a particular observation by the external observers.

In general, PCTL formulas must always evaluate to a Boolean value, the probabilistic operators $\mathcal{P}$ should always include a bound $\bowtie\ p$. However, it is often useful to know the actual probability that some behavior is observed, rather than just check that the probability is above or below a given bound. PRISM allows properties of the form $\mathcal{P}_{=?}[\Psi]$. These formulas return a numerical rather than a Boolean value. Again note that, for MDPs, since probabilities can only be computed once the nondeterministic choices have been resolved. Hence, there is actually a *minimum* and a *maximum* probability of a path formula being satisfied, quantifying over all possible resolutions. Therefore, for MDPs PRISM allows two possible types of formula: $\mathcal{P}_{max=?}[\Psi]$ and $\mathcal{P}_{min=?}[\Psi]$, which return the maximum and minimum probabilities, respectively.